UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/751,190	12/29/2000	Danny Hendler	6599P003X6	8523

8791 7590 02/11/2004

BLAKELY SOKOLOFF TAYLOR & ZAFMAN
12400 WILSHIRE BOULEVARD, SEVENTH FLOOR
LOS ANGELES, CA 90025

EXAMINER	
PATEL, HARESH N	

ART UNIT	PAPER NUMBER
2154	22

DATE MAILED: 02/11/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/751,190	HENDLER ET AL.	
	Examiner	Art Unit	
	Haresh Patel	2154	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☐ Responsive to communication(s) filed on _____.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-27 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-27 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-27 are presented for examination.

Oath/Declaration

2. Signature of the fifth inventor (Shmuel Melamed) is missing.

Drawings

3. Figure 2 has been replaced with a same figure contents and designated as --Prior Art--.

Response to Arguments

4. Applicant's arguments with respect to claims 1-27 have been considered but are moot in view of the new ground(s) of rejection.

Claim Rejections - 35 USC § 112

The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

5. Claims 2, 5, 9, 10, 13, 14 and 19, 22 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. The claim(s) contains subject matter, which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention.

Claim 2 recites the term "the method further comprises allocating storage space at the client device for the second archive based on size information in the end-of-central-directory record". There is insufficient antecedent basis for this limitation in the claim.

Claim 2 recites the term "a first one of the streamed modules comprises a ZIP file end-of-central-directory record". There is insufficient antecedent basis for this limitation in the claim.

Claim 5 recites the term "a first one of the streamed modules comprises a Java class file". There is insufficient antecedent basis for this limitation in the claim.

Claim 9 recites the term "wherein the method further comprises: at the client device, processing a request from the executing application to access a file in a first one of the modules: and determining if the first module is present in the second archive; sending streaming control data to the server to request the first module when the first module is not present in the second archive, and streaming the first module from the server to the client device in response to receipt of said streaming control data at the server". There is insufficient antecedent basis for this limitation in the claim.

Claim 10 recites the term "wherein streaming the first module in response to the receipt of said streaming control data comprises interrupting streaming of another one of the modules". There is insufficient antecedent basis for this limitation in the claim.

Claim 13 recites the term "the second archive file comprises a different arrangement of the streamed files than in the first file, said different arrangement being functionally equivalent to an arrangement of files in the first archive file". There is insufficient antecedent basis for this limitation in the claim.

Claim 14 recites the term "the first archive comprises at least one file comprising non-executable data and at least one file comprising executable program code". There is insufficient antecedent basis for this limitation in the claim.

Claim 19 recites the term "wherein streaming comprises: receiving a first module in the module set while an application is executing; storing the first module on local storage media at the second computer; and wherein the method further comprises integrating the first module with the executing application". There is insufficient antecedent basis for this limitation in the claim.

Claim 19 recites the term "the instructions to construct in accordance with ZIP file format specifications further comprise instructions to position received ones of the modules in the second archive in an order of receipt of the modules and instructions to alter data in the second archive's central directory header to indicate the placement of the ones of the modules in the second archive". There is insufficient antecedent basis for this limitation in the claim.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claims 1, 11, 12, 20, 24 and 27, are rejected under 35 U.S.C. 103(a) as being unpatentable over Rodriguez et. al. 6,427,149 (Hereinafter Rodriguez) in view of Distributed Systems Concepts and Design, pages 69 – 73, 99 - 101, 1995, Coulouris et. al (Hereinafter Coulouris).

7. As per claims 1, 11, 12, 20, 24 and 27, Rodriguez teaches the following:

a method of streaming an archive file from a server to a client device, the method comprising,

a computer-implemented method of transmitting data in an archive file from a server device to a client device, the client device comprising an execution environment configured to provide ones of a collection of logically separate files in a received archive file to an executing application in an execution-time determined order, the method comprising,

a data storage apparatus comprising instructions to configure a computerized device to,

a data storage apparatus comprising instructions to configure a computer to,

a system for transferring information modules between computers, the system comprising:

at a server, extracting ones of a plurality of modules from a first archive file, extracting ones of a collection of logically separate files from a first archive file, automatically extract a collection of logically separate modules from a first archive file, means for extracting a collection of modules associated with the application from an archive file (e.g., block 47, extract individual compressed files and transmit to client computer at web server, figure 3, the web server extracts only the selected files for transmission to the client computer, abstract);

transmitting the extracted modules from the server to the client device, transmitting the extracted ones of the separate files from the server device to the client device, transmit the extracted modules to a client device, means for transferring the selected sequences from the second computer to the first computer (e.g., The web server responds by executing (47) the file

extraction utilities to extract and optionally decompress the selected files from the archive file. Those extracted files are then transmitted (48) to the client computer or network client, where they are saved (49) and/or decompressed (50) by the appropriate software application. The time (45) spent waiting for the transfer to complete using this method is substantially reduced compared to the conventional method as the total data volume to be downloaded is reduced, and as unwanted data is not downloaded at all, col., 5, lines 1- 54),

receiving the streamed modules at the client device, receive archive structure data from another computer, the archive structure data specifying a structure of a first archive file, receive from the other computer ones of the plurality of modules derived from the first archive file; in accordance with the archive structure data, receiving the streamed files at the client device (e.g., block 45, wait for extracted ZIP at a client computer, figure 3);

automatically constructing a second archive file at the client device, the second archive file comprising the received modules, construct a second archive file from the received ones of the plurality of modules to produce a second archive file functionally equivalent to the first archive file, means for receiving a sequence of modules associated with the application and constructing an archive file, constructing a second archive files at the client device from the received files (e.g., block 49, save ZIP file and/or launch ZIP utility directly); and

providing data from at least one of the received modules in the second archive to an executing application, means for executing an application, an archive file comprising the received sequence of modules while the application is executing, means for integrating a first module in the constructed archive file with the application (e.g., for files which were selected to be extracted and decompressed by the web server prior to transmission to the client computer,

the proper application program on the client computer which is registered to that file type may be optionally launched to allow immediate operation on the file after it is received. For example, if the file is an Apple Quicktime video clip, the proper Apple viewer could be launched. Or, if the file were a Lotus WordPro word processor document, Lotus WordPro would be launched. The method of using file name extensions to associate files with and cooperatively launch the application is well understood in the art, and similar functionality is found in most personal computer operating systems, col., 5, lines 1- 54).

Rodriguez teaches transmitting of any files including achieve files, video clips from a server to the client.

However, Rodriguez does not specifically mention about the term "streaming" for transmission.

Coulouris teaches the following:

streaming (e.g., the size of TCP streams is unlimited, so the TCP transport layer protocol must decompose the stream of data supplied to it by application programs into chunks of data and construct IP packets that are not more than 64K kilobytes in length, page 72, Stream communication is used to implement some services such as remote login and file-transfer in networked environments, pages 99-101).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teachings of Rodriguez with the teachings of Coulouris to facilitate a communication mechanism to transfer the data modules from one computer to the another. The motivation would be obvious because stream communication is supported by buffering which enables the sender to get ahead of the recipient, as suggested by Coulouris.

8. Claims 2, 21 and 22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rodriguez and Coulouris in view of Bittinger et. al. 6,148,340 (Hereinafter Bittinger).

9. As per claims 2, 21 and 22, Rodriguez teaches the following:

the first archive file (e.g, ZIP file),

zip file modules (e.g., including a header section (2), the compressed data blocks (3), and a tail section (4), col., 2, lines 47 – 65).

However, Rodriguez and Coulouris do specifically mention about an end-of-central-directory record and the Java Archive file.

Bittinger teaches the following:

the first archive file is a Java Archive file (e.g., Many software manufacturers make non-applet software available on the world wide web (the Web) in the form of ZIP or self-extracting ZIP archive files. ZIP archive files are container files that have a publicly known architecture.. JAR files are actually a subset of the overall ZIP file architecture. The ZIP files can be downloaded directly with a Web browser, but such downloads take a significant amount of time. Some companies have created products that will compare a file with a previous version of the file prior to refreshing the file on the remote computer in an effort to reduce the response time as perceived by the user. An example of such a product is Novadigm's EDM, col., 1, line 55 – col., 2, line 13),

a first one of the streamed modules comprises a ZIP file end-of-central-directory record and the method further comprises allocating storage space at the client device for the second archive based on size information in the end-of-central-directory record, the archive structure

data comprises a ZIP file central directory record; and the instructions to construct the second archive file comprise instructions to position the received central directory record in the second archive in accordance with ZIP file format specifications, the instructions to construct in accordance with ZIP file format specifications further comprise instructions to position received ones of the modules in the second archive in an order of receipt of the modules and instructions to alter data in the second archive's central directory header to indicate the placement of the ones of the modules in the second archive (e.g., block 505, central directory record, figure 5, When processing a JAR or ZIP file, the present invention constructs a catalog from the central directory and the object headers, each object in the container has a corresponding entry in the catalog which includes the member name (and path), size, a 32 bit CRC(cyclic redundancy check) of the object contents as well as the size and CRC of the object header. In the present invention, upon receiving a container for the first time, the SSI constructs and caches its catalog, col., 3, line 20 – col., 5, line 35).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teachings of Rodriguez and Coulouris with the teachings of Bittinger to facilitate a first data packet containing the information of the number of packets of a transmitted zip file message. The motivation would be obvious because a central directory header information is useful to know how many packets needs to be received to construct an achieve file and to know what size of memory needs to be allocated for the message, as suggested by Bittinger. The motivation for usage of a JAR file would be obvious because ZIP archive files are container files that have a publicly known architecture and JAR files are actually a subset of the overall ZIP file architecture, as suggested by Bittinger.

10. Claim 3 is rejected under 35 U.S.C. 103(a) as being unpatentable over Rodriguez, Bittinger and Coulouris in view of Slaughter et. al. 6,643,650 (Hereinafter Slaughter)

11. As per claim 3, Rodriguez, Bittinger and Coulouris do not specifically mention about the Java Micro Edition execution environment.

Slaughter teaches the following:

the client device comprises a Java Micro Edition execution environment (e.g., In one embodiment, message capable network layer 106 may be implemented from the networking classes provided by the Java2 Micro Edition (J2ME) platform, col., 2, line 15 – col., 6, line 55).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teachings of Rodriguez, Bittinger and Coulouris with the teachings of Slaughter to facilitate a client device having a Java Micro Edition execution environment. The motivation would be obvious because the Java2 Micro Edition platform is suitable for devices that do not have the resources for a full Java platform or in which it would not be efficient to run a full Java platform. Since J2ME already provides a message capable family of networking protocols (to support sockets), it follows that for the small footprint cost of adding messaging layer 104, distributing computing facilities may be provided for small devices that already include J2ME, as suggested by Slaugher.

12. Claims 4, 9, 15, 16, 19, 23, 25 and 26 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rodriguez and Coulouris in view of White et. al. 6,230,184 (Hereinafter White).

13. As per claims 4, 9, 15, 16, 19, 23, 25 and 26, Rodriguez and Coulouris do not specifically mention about predicting an order of utilization of modules at the client device.

White teaches the following:

the criteria comprises data stored at the first computer in a streaming control database (e.g., on server 751 in data store 775, figure 7),

receiving a request at the first computer from the second computer (e.g., When a client computer issues a request to server 751 for the optimized program 725 the optimized file 730 is sent first through the network 760, figure 7),

modifying the predictive data stored in the streaming control database based on the request, streaming in accordance with predetermined criteria predicting an order of utilization of modules at the client device, processing a request from the executing application to access a file in a first one of the modules: and determining if the first module is present in the second archive; sending streaming control data to the server to request the first module when the first module is not present in the second archive, and streaming the first module from the server to the client device in response to receipt of said streaming control data at the server, ordering the separate files extracted from the archive file based on criteria predicting an order of utilization of the separate files at the second computing device; and streaming the separate files from the first computer to the second computer in accordance with the ordering, receiving a first module in the module set while an application is executing; storing the first module on local storage media at the second computer; and wherein the method further comprises integrating the first module with the executing application, process a request from an executing application to access data in a first one of the modules; determine if the first module is present in the second archive; send control

data to the other computer to request the first module when the first module is not present in the second archive (e.g., Optimally Executing a Computer Program, Embodiments of the invention can be used to automatically determine the files needed to optimally execute a computer program until a desired state is reached. In one embodiment, the invention automatically creates an optimized file containing the files that are necessary for the computer program to reach the desired state. Creating an optimized file is useful, for example, when a computer program is transported across a network to another computer where it is then executed. In one embodiment of the invention, the files needed to execute the computer program to a desired state are sent across the network to a client computer in the optimized file. The desired state may be, for example, the point at which the computer program is capable of receiving user input. Once the files contained in the optimized file are received, the client computer can execute the program to the desired state without waiting for the remaining files used during the execution of the program to be transmitted across the network. Since the program can begin execution before all the files that comprise the computer program are received, the amount of time required to start the program is reduced, col., 9, lines 21 – 44).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teachings of Rodriguez and Coulouris with the teachings of White to facilitate a predicting mechanism to provide necessary data packets at the client device. The motivation would be obvious because the server can send the necessary files required to the client using automatic determination mechanism. This would avoid not sending unnecessary files to the client and would help optimally execute a computer program, as clearly suggested by White.

14. Claims 5-8 is rejected under 35 U.S.C. 103(a) as being unpatentable over Rodriguez and Coulouris in view of Bittinger, White and "Official Notice".

15. As per claims 5-8, Rodriguez teaches the following:

the first archive file (e.g, ZIP file),

zip file modules (e.g., including a header section (2), the compressed data blocks (3), and a tail section (4), col., 2, lines 47 – 65).

However, Rodriguez and Coulouris do specifically mention about the Java Archive file.

Bittinger teaches the following:

the first archive file is a Java Archive file (e.g., Many software manufacturers make non-applet software available on the world wide web (the Web) in the form of ZIP or self-extracting ZIP archive files. ZIP archive files are container files that have a publicly known architecture. JAR files are actually a subset of the overall ZIP file architecture. The ZIP files can be downloaded directly with a Web browser, but such downloads take a significant amount of time. Some companies have created products that will compare a file with a previous version of the file prior to refreshing the file on the remote computer in an effort to reduce the response time as perceived by the user. An example of such a product is Novadigm's EDM, col., 1, line 55 – col., 2, line 13).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teachings of Rodriguez and Coulouris with the teachings of Bittinger to facilitate an achieve file containing compressed files. The motivation for usage of a JAR file would be obvious because ZIP archive files are container files that have a publicly known

architecture and JAR files are actually a subset of the overall ZIP file architecture, as suggested by Bittinger.

Rodriguez, Bittinger and Coulouris do specifically mention about the other details of the client computer and usage of digital signature to validate received files.

White teaches the following:

a first one of the streamed modules comprises a Java class file (e.g., figure 3, distribution program consisting of multiple files sent to the client using achieve file, col., 7, lines 1-63),

the executing application comprises a Java application (e.g., A program written in Java that executes without the aid of a web browser is called a Java application, col., 1, line 6 – col., 7, line 30) and

providing data from at least one of the stored file to the executing application comprises dynamically loading the Java class file for access by the Java application (e.g., When a program is loaded using a standard "class loader" the contents of the files that are loaded to run the application are stored in a hash table. The standard "class loader" checks the hash table before it loads a new file. If the file name is already in the hash table it is not loaded. This prevents loading duplicate versions of the same file, It should be apparent to one skilled in the art that the files placed into the optimized file or any other type of file can be any of the files used by the program during execution. Further, the desired state of the computer program can be defined to be any point during the execution of the computer program, col., 4, line 15 – col., 8, line 25),

at least one of the modules in the Java Archive file comprises Java Archive meta information comprising an identification of files in other ones of the modules; and the method further comprises streaming the meta- information from the server to the client device and

validating a file in a received modules using the meta-information, validating comprises comparing a file name and path associated with a file received in a streamed module to a file name and path specified in the meta- information, the meta-information comprises a first digital signature and the method further comprises: computing a second digital signature based on data in a first one of the received modules; and comparing the second digital signature to the first digital signature to determine validity of a file received in a streamed module, (e.g., At step 906, optimizor 750 obtains a manifest file. In one embodiment of the invention, the manifest file is placed in a directory called META-INF. The manifest file may also include other arbitrary information. At step 907, the optimizor 750 determines whether it is desirable to digitally sign the files listed in intermediate file 723. If the optimizor determines to digitally sign a file, step 908 is executed and information about the digital signature is placed into the directory named META-INF. In one embodiment of the invention, step 909 is also executed and the digital signature information is placed in the manifest file. Otherwise, the optimizor 750 proceeds to step 910 where all the files listed in intermediate file 723 are combined into a single file called the optimized file 730. Once these steps are complete the optimizor 750 outputs the optimized file 730 to a storage medium, col., 9, line 23 – col., 15, line 47).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teachings of Rodriguez, Bittinger and Coulouris with the teachings of White to facilitate a Java environment at the client computer and a mechanism to validate received files using digital signature. The motivation would be obvious because unlike many programming languages, in which a program is compiled into machine-dependent, executable program code, programs written in the Java programming language are compiled into machine-

independent bytecode classfiles. Each classfile contains code and data in a platform-independent format called the classfile format. The computer system acting as the execution vehicle contains a program called a virtual machine, which is responsible for executing the bytecode. The virtual machine provides a level of abstraction between the machine-independent bytecode classes and the machine-dependent instruction set of the underlying computer hardware. Virtual machines exist for a variety of different operating systems. A "class loader" within the virtual machine is responsible for loading the bytecode classfiles as needed, and either an interpreter executes the bytecodes directly, or a "just-in-time" (JIT) compiler transforms the bytecodes into machine code, so that they can be executed by the processor, as suggested by White. The JAR format is a set of conventions for associating digital signatures, and other information with the files held in the JAR file. Hence, implementation of the digital signature would help a client to utilize a received JAR file, as suggested by White.

Rodriguez, Bittinger, White and Coulouris do not specifically mention about minor details of the digital signature implementation, i.e., computing of the digital signature. "Official Notice" is taken that both the concept and advantages of providing the specific way of computing of the digital signature is well known and expected in the art and would be an obvious design choice to include specific way of computing digital signature.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to include minor details of the way the digital signature is calculated with the teachings of Rodriguez, Bittinger, White and Coulouris to facilitate validating of received files. The JAR format is a set of conventions for associating digital signatures, and other information

with the files held in the JAR file. Hence, implementation of the digital signature would help a client to utilize a received JAR file, as suggested by White.

16. Claim 10 is rejected under 35 U.S.C. 103(a) as being unpatentable over Rodriguez, Coulouris and White in view of "Official Notice".

17. As per claim 10, Rodriguez, White and Coulouris do not specifically mention about interrupting streaming of another one of the modules.

"Official Notice" is taken that both the concept and advantages of providing interruption of streaming of another one of the modules is well known and expected in the art.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to include interruption of streaming of another one of the modules with the teachings of Rodriguez, White and Coulouris to facilitate stopping of the current transmission of the data being sent and to send the newly requested data needed by the client device. This would help to not send unwanted data from server to the client.

18. Claim 13 is rejected under 35 U.S.C. 103(a) as being unpatentable over Rodriguez and Coulouris in view of Schmidt et al 6,535,894.

19. As per claim 13, Rodriguez and Coulouris do not specifically mention that the second archive file comprises a different arrangement of the streamed files than in the first file.

Schmit teaches the following:

the second archive file comprises a different arrangement of the streamed files than in the first file, said different arrangement being functionally equivalent to an arrangement of files in

the first archive file (e.g., According to aspects of the present invention, an original archive file having one or more entries is created, where each entry in the original archive file is itself a file, and where each entry in the archive file may comprise any file type, including an archive file. The original archive file is transmitted to a client computer. Subsequently, a target archive file is created, wherein one or more of the entries in the target archive file are typically expected to be identical to one or more entries in the original archive file. Given the original archive file and the target archive file, a difference archive file is created. The difference archive file comprises an index file describing the changes between the original archive file and the target archive file, and also comprises a set of entries corresponding to the entries in the target archive file that are not contained in the original archive file. The difference archive file is transmitted to the client computer, instead of requiring that the entire target archive file be transmitted. At the client computer, the difference archive file is applied to the original archive file to produce a synthesized archive file, wherein the synthesized archive file is functionally identical to the target archive file, and wherein each entry in the synthesized archive file is identical to a corresponding entry in the target archive file, abstract).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teachings of Rodriguez and Coulouris with the teachings of Schmit to facilitate a mechanism to construct the archive file contents from the streamed files similar to the original archive file contents. The file contents of the constructed archive file necessary for the client device would support the client application regardless of the arrangement of the files in the archive.

20. Claim 14 is rejected under 35 U.S.C. 103(a) as being unpatentable over Rodriguez and Coulouris in view of "Official Notice".

21. As per claim 14, Rodriguez teaches the following:

the first archive comprises non-executable data and executable program code (e.g., archive file to be viewed prior to downloading any compressed data, such that individual files can be selected and downloaded, col., 2, line 25 – col., 5, line 28).

However, Rodriguez and Coulouris do specifically mention about the first archive comprises at least one file comprising non-executable data and at least one file comprising executable program code. "Official Notice" is taken that both the concept and advantages of providing one file comprising non-executable data and at least one file comprising executable program code is well known and expected in the art and would be an obvious design choice to include just one file comprising non-executable data and at least one file comprising executable program code.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to include just one file comprising non-executable data and at least one file comprising executable program code with the teachings of Rodriguez and Coulouris to facilitate a combination of one file comprising non-executable data and at least one file comprising executable program code. An archive file containing whatever files needed by the user would be transmitted to the user, as suggested by Rodriguez.

22. Claims 17 and 18 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rodriguez, White and Coulouris in view of Kodialam and Patel et al. Pub. No. US 2002/0157089 A1, Oct. 24, 2002 (Hereinafter Patel).

23. As per claims 17 and 18, Rodriguez, White and Coulouris do not specifically mention about transition records associating weighted values with execution transitions between selected files in the collection.

Patel teaches the following:

transition records associating weighted values with execution transitions between selected files in the collection (e.g., First of all, frequently used file blocks can be streamed to the client machine before other less used file blocks. A frequently used file block is cached locally on the client cache before the user starts using the streamed application for the first time. This has the effect of making the streamed application as responsive to the user as the locally installed application by hiding any long network latency and bandwidth problems, paragraph, 0307).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teachings of Rodriguez, White and Coulouris with the teachings of Patel to facilitate streaming base on most frequently files that are used by several users. The motivation would be obvious because it would be beneficial by making the streamed application as responsive to the user as the locally installed application by hiding any long network latency and bandwidth problems, as clearly suggested by Patel.

Rodriguez, White, Patel and Coulouris do not specifically mention about streaming order based on a path algorithm.

Kodialam teaches the following:

ordering the separate files comprises ordering based on transition record values, processing transition record information using a path determination algorithm (e.g., a set of preferred paths through the network nodes, and may use the weights to calculate the set of preferred paths. Each preferred path has a minimum total weight between nodes as well as minimum summed weight through nodes of the path, which is known in the art as shortest-path-algorithm, col., 1, line 34 – col., 4, line 22).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teachings of Rodriguez, White, Patel and Coulouris with the teachings of Kodialam to facilitate streaming base on the shortest routing path between the communicating computers. The motivation would be obvious because it would be beneficial by making the streamed application as responsive to the user as the locally installed application by hiding any long network latency and bandwidth problems, as clearly suggested by Kodialam.

Conclusion

24. This application is a continuation in part of application number 09/120,575, which does not teach the entire limitations of any of the independent claims.

25. Examiner also makes a note that the claims are not completely identical to the claims of the patent number 6,311,221, but are very close for not patentably distinct from each other.

26. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Haresh Patel whose telephone number is (703) 605-5234. The

examiner can normally be reached on Monday, Tuesday, Thursday and Friday from 10:00 am to 8:00 pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, John Follansbee, can be reached at (703) 305-8498.

The appropriate fax phone number for the organization where this application or proceeding is assigned is (703) 872-9306.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

Haresh Patel
February 3, 2004

JOHN FOLLANSBEE
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100